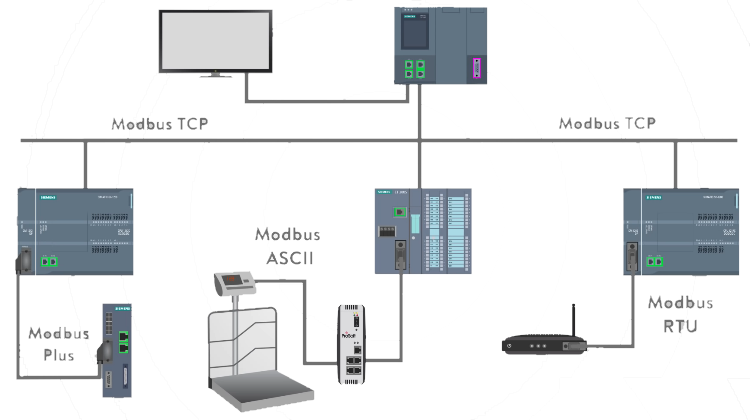
**Notes – Modbus (**[**here**](https://www.youtube.com/watch?v=JBGaInI-TG4)**)**

Oldest and most popular communications protocol. Common language for devices and equipment.

There exist a Modbus organization to manage the protocol.



Most common versions of Modbus include:

**Modbus RTU**: remote terminal unit. A serial protocol; RS-232/485, 16-bit CRC. Master/Slave architecture (aka client/server). Due to the simplicity of the messages, the basic 16-bit Modbus RTU register structure can be used to pack in floating point, tables, ASCII text, queues, and other unrelated data.

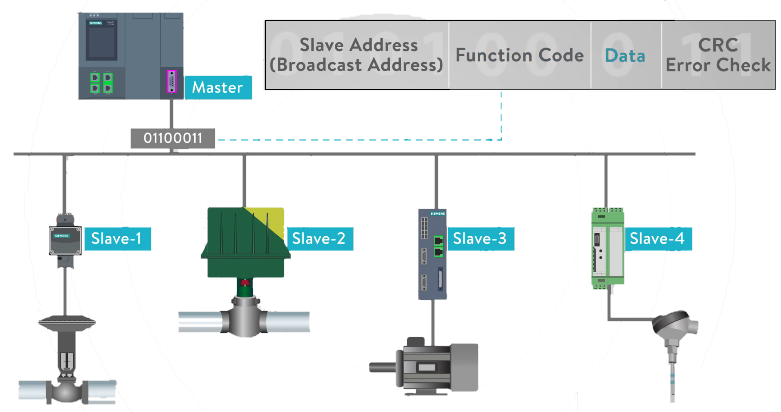
**Modbus ASCII**: ASCII mode uses ASCII characters to begin and end messages whereas RTU uses time gaps (3.5 character times) of silence for framing. The two modes are incompatible so a device configured for ASCII mode cannot communicate with one using RTU. Modbus ASCII messages require twice as many bytes to transmit the same content as a Modbus RTU message.

**Modbus Plus**: Modbus Plus is a local area network (LAN) system that allows networked devices to exchange messages to monitor and control processes in industrial plants. It supports up to 64 devices (nodes) and transfers data at 1 million bits per second (1Mbaud). Modbus Plus is a peer-to-peer protocol that indicates both the hardware and software layers. It differs from Modbus, which is a serial master/slave protocol that can work through any serial type media.

**Modbus TCP**: runs on Ethernet physical layer. Modbus TCP uses TCP/IP and Ethernet to transport Modbus message structure data between devices. It combines a physical network (Ethernet), a networking standard (TCP/IP), and a standard method of representing data (Modbus). Modbus defines the address of each data element as ranging from 0 to 65,535. However, each data element is numbered from 1 to n, where n has a maximum value of 65,536. There is no set limit to the number of masters, but the slaves can only answer one master at a time.



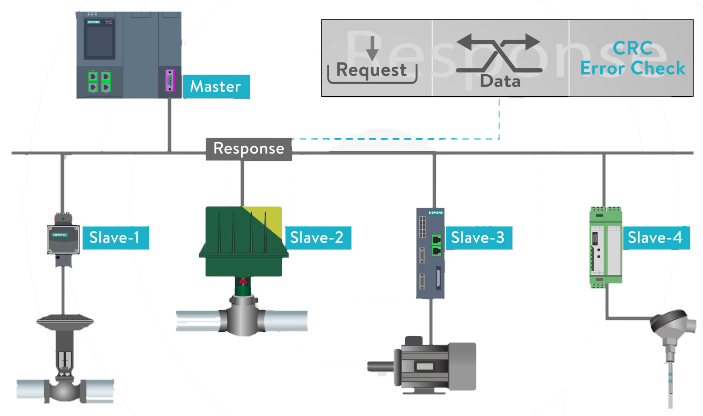
Modbus is able to function on both point-to-point and multidrop networks.



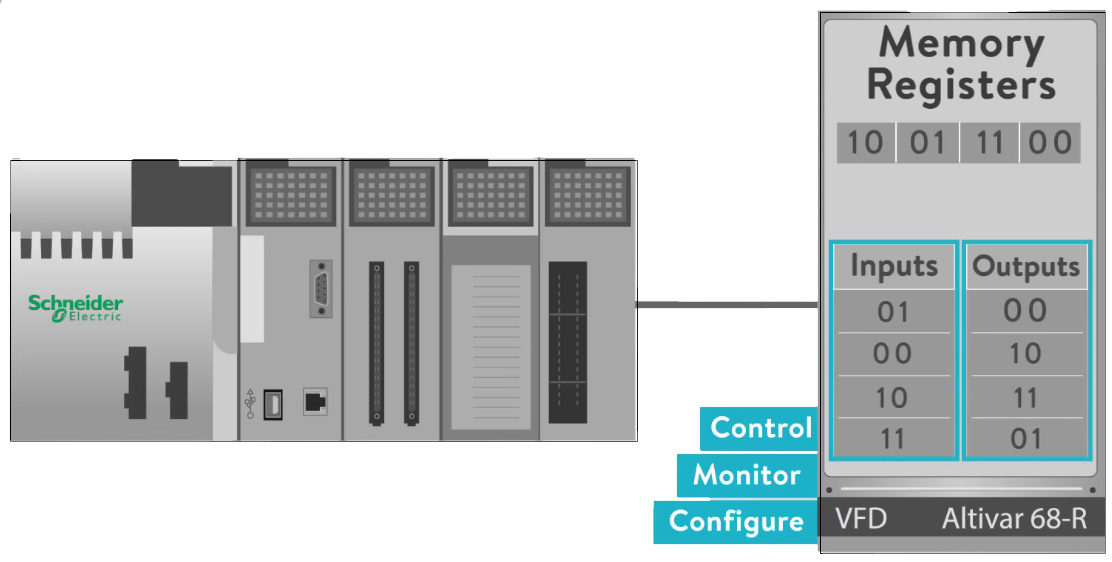
The above represents a master query consisting of a slave address, function code, data, and crc.

Each Modbus message uses this same message structure, the sequence and order are the same.

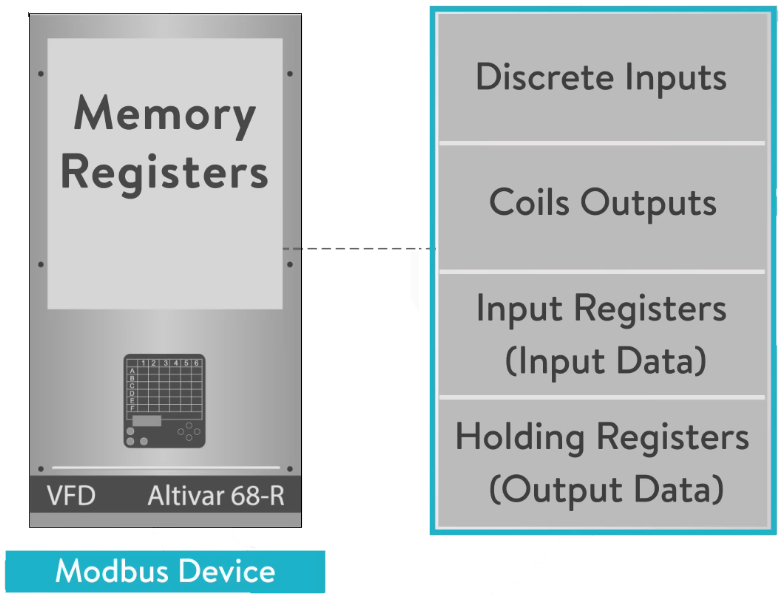
Master always starts the conversation.



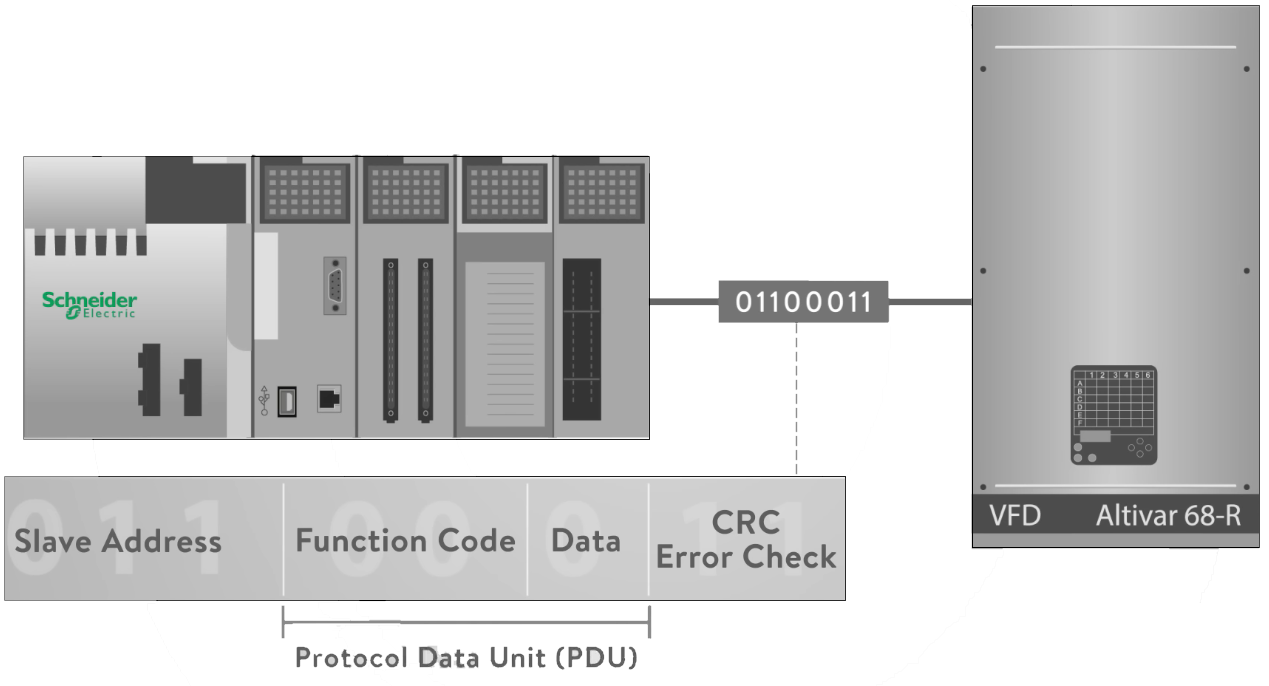
The salve’s response consist of fields confirming it received the request; they include request, data, and crc. Bad crc results initiate an exception message.



Modbus functions preform, read, and write instructions to the slaves internal memory registers to configure, monitor, and control the slaves inputs and outputs.

Modbus devices will typically include a register map outlining where the configuration, input, and output data can be written and read from.

The table at left shows the Modbus data model consisting of 4 basic data types.



The service request area of the message consisting of function code and number of data bytes requested by the master, is also known as Protocol Data Unit (PDU).

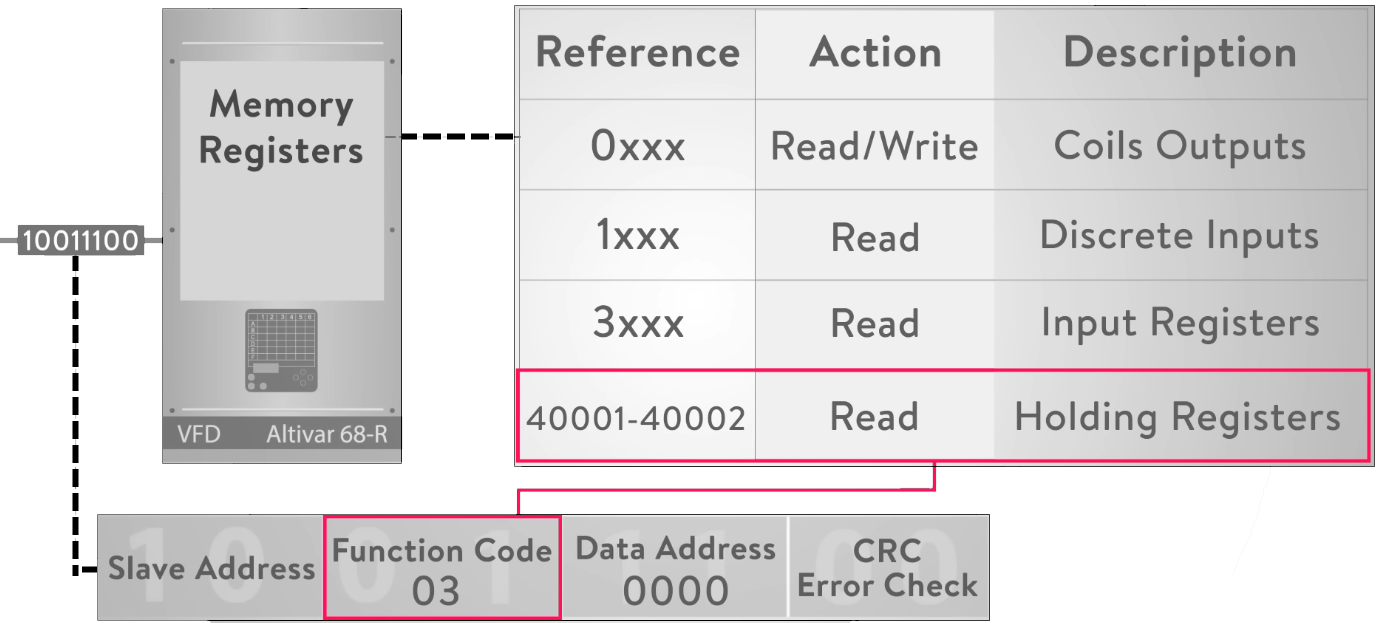
A

The Modbus memory registers of a device are organized around the four basic data reference types and this data type is further identified by the leading number used in the devices memory address.

For example; a zero based register referencing a message to read or write discrete outputs or coils.

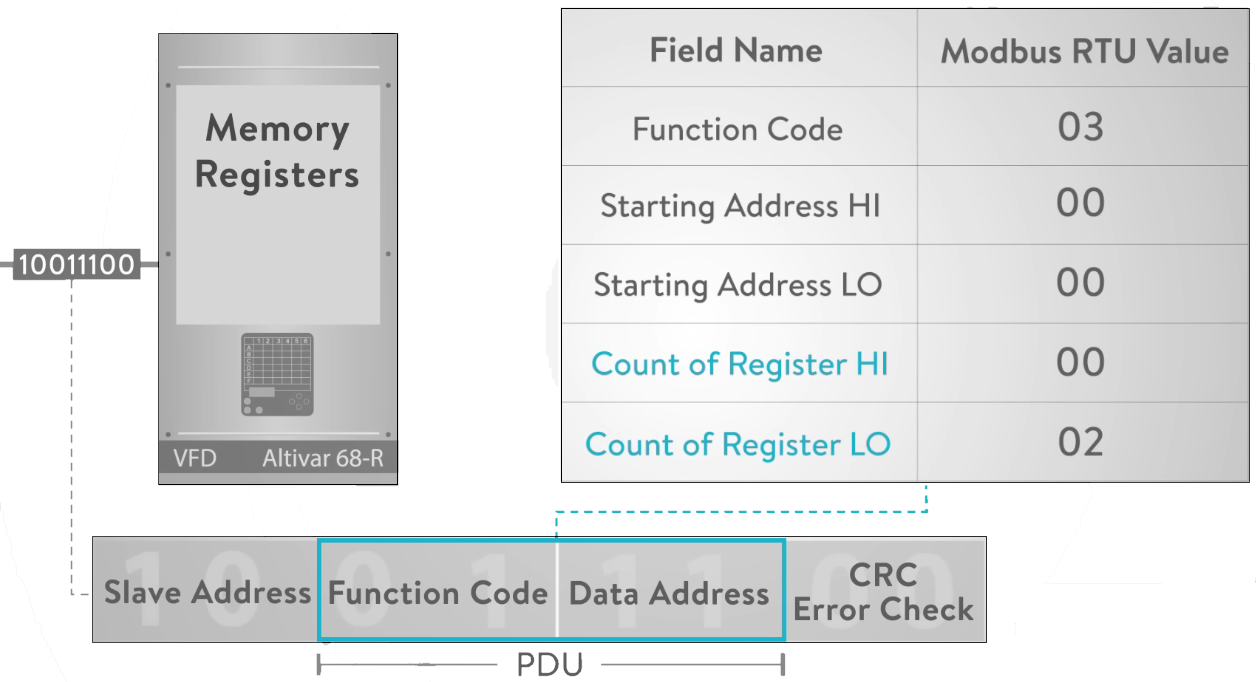
Or a one based register referencing reading discrete inputs. Or a three based register referencing reading input registers. Or a four based register referencing to reading or writing output or holding registers.



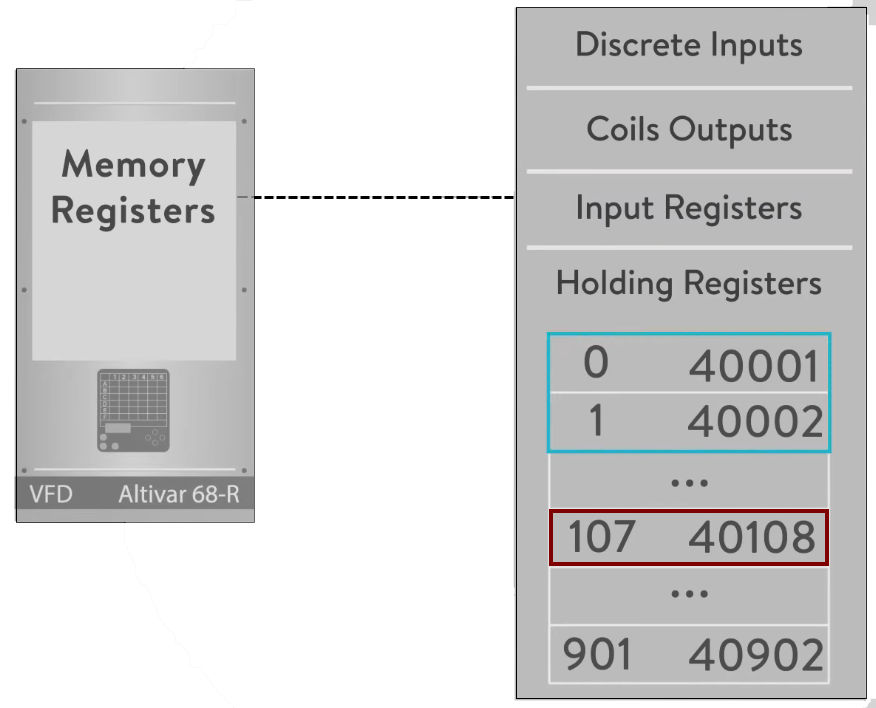


The function code field specifies which register data group it reads or writes to and from the slave.

For example: a function code “03” reads holding registers 40001 to 40002 is addressed as a data register 0000 in the data address field of the message sent to the slave. The function code “03” works on this holding register type in the slave’s data map because the request specifies using a holding register data type operation. This type of addressing in the request is implied

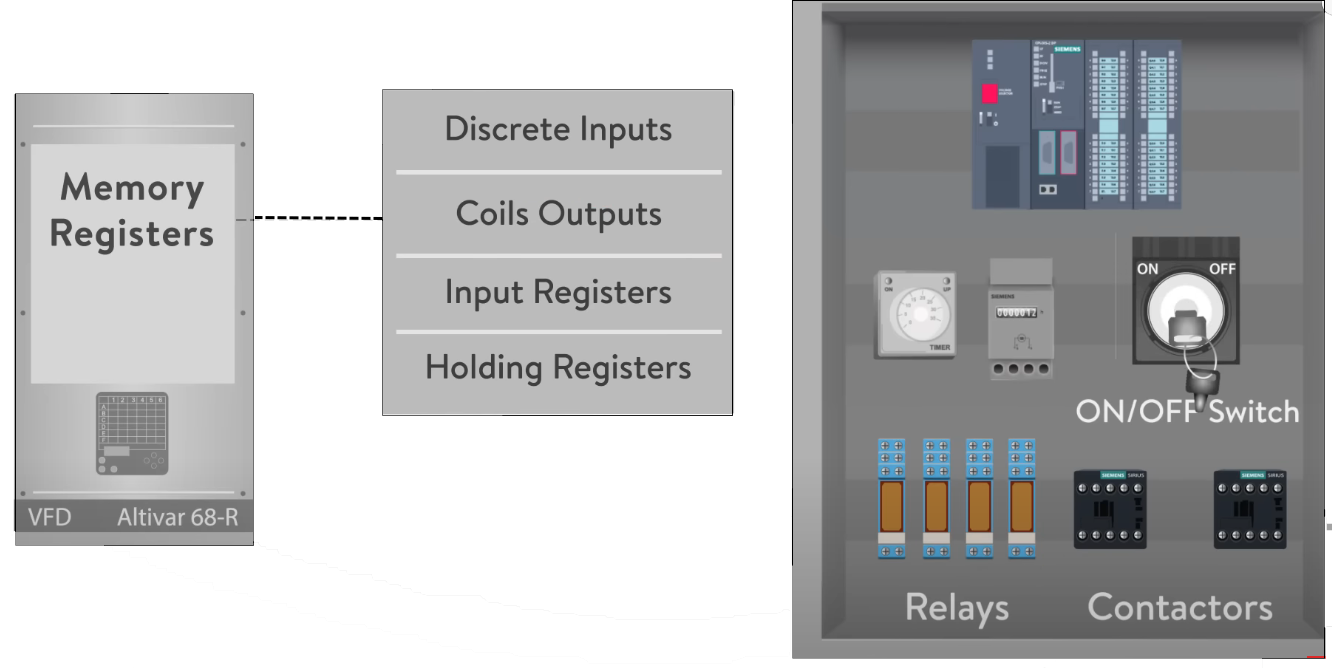


The fields in the PDU are broken down into bytes and grouped by the field name. The request message contains the function code of 03, the starting address of 00 in both hi and lo, and the count number of addresses to read from the slave. Register hi and lo bytes of 0002 specifies the starting register and quantity of registers to be read from the slave.

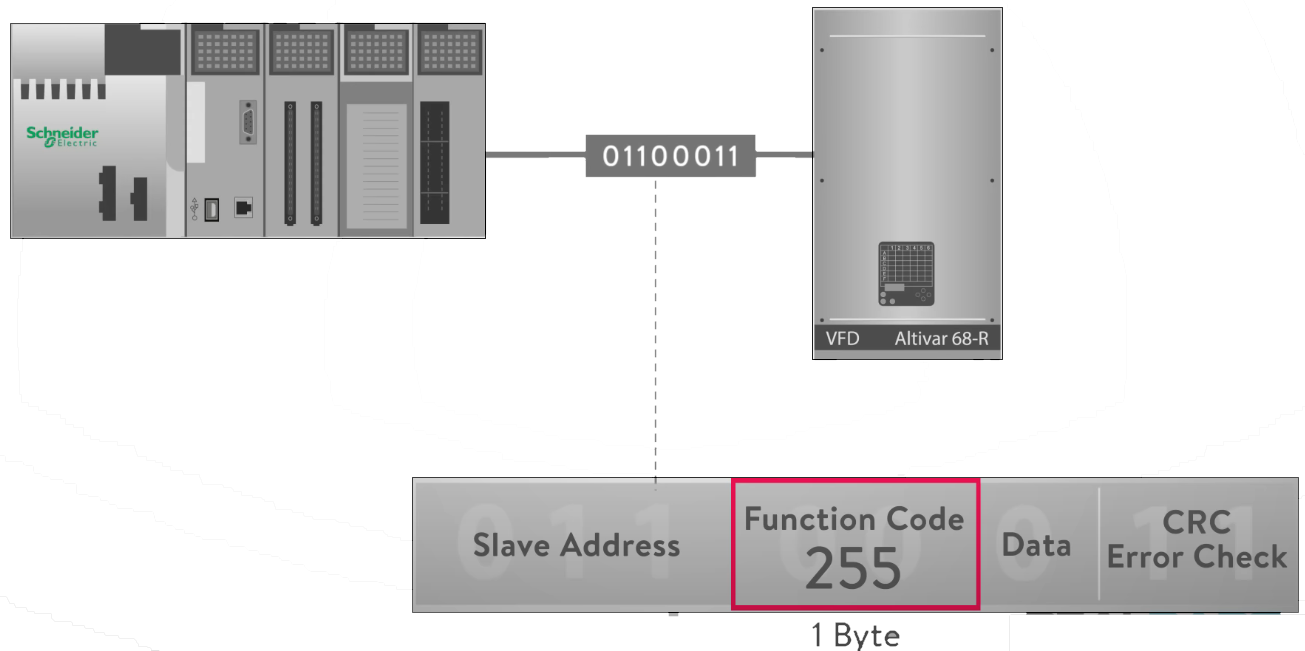


Example: a request to read the first two registers in the holding register area, 0 to 1, register 40001 to 40002.

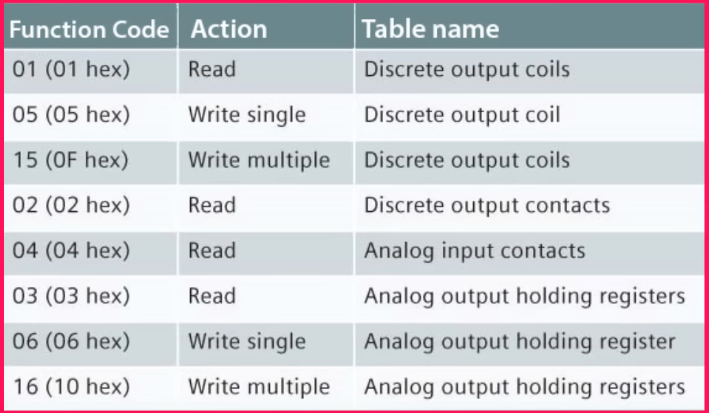
Example: the holding register 40108 is actually addressed as register 107 in the message data area of the PDU.



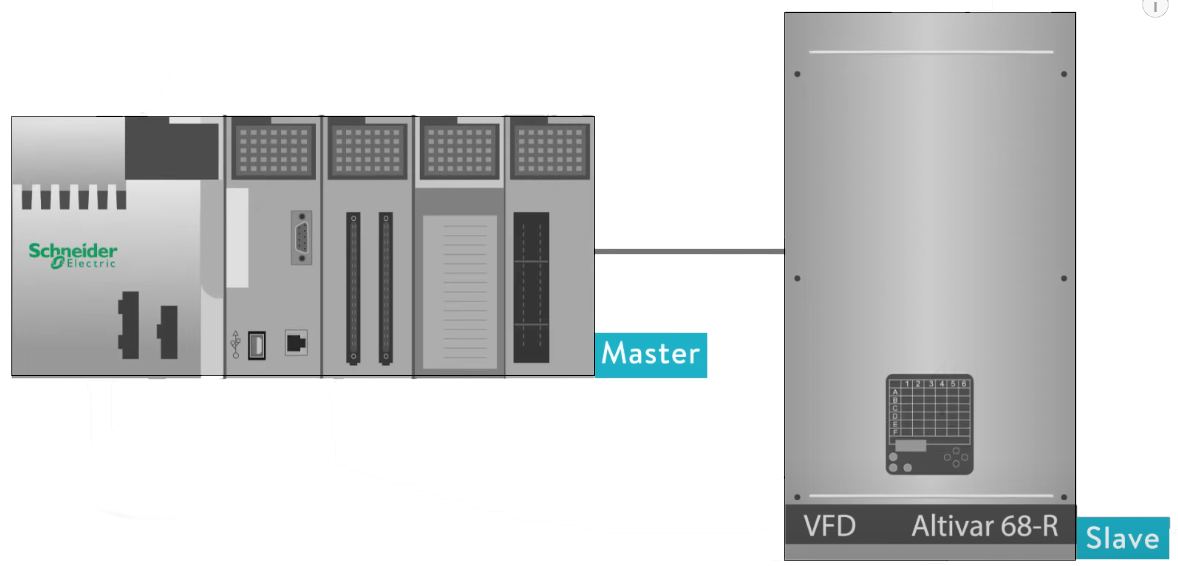
Many of the data types are named from the view of the driving relay. For example, a single bit physical output is called a coil and a single bit physical input is called a discrete input or contact.



The function field of the message will contain one byte which tells the slave what action to take. The range of valid function codes is from 1 to 255.

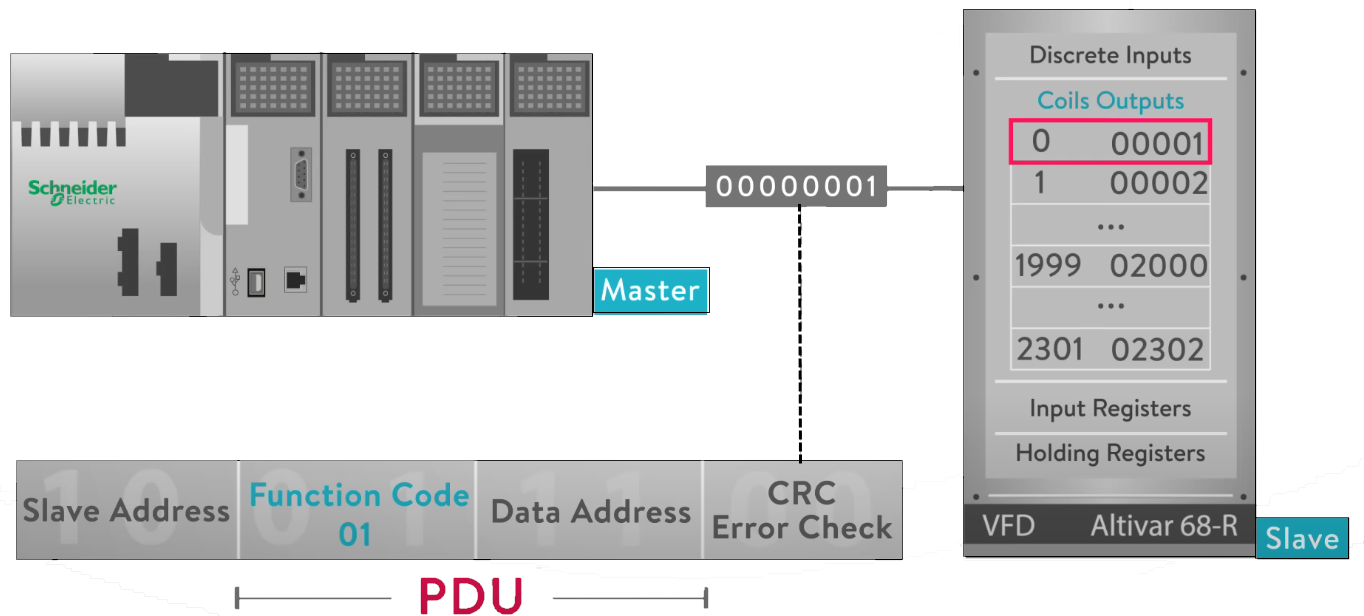


This table highlights a subset of standard Modbus functions.

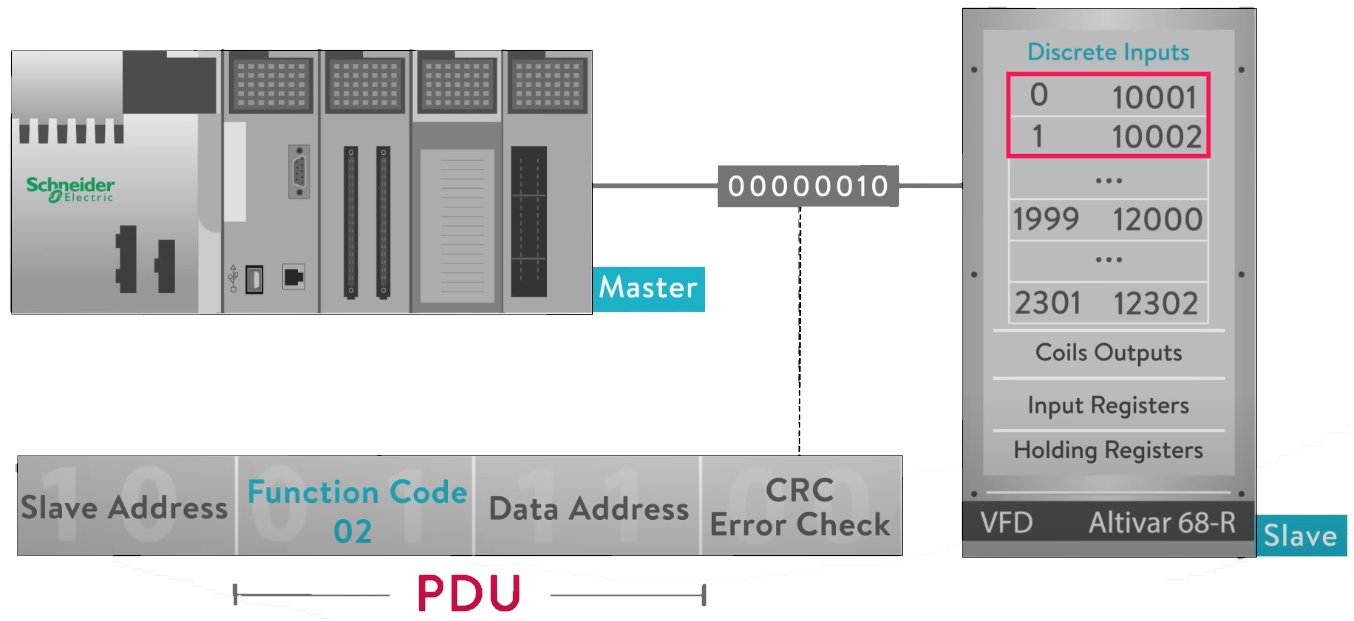


In addition, the master request data field provides the salve with any additional information required by the slave to complete the action specified by the function code in the masters request. Typically including the slave map register address, the number of registers to provide in the request, and any write data from the master.

The slaves normal response simply echoes the function code of the original request. But the salves error response returns a code that is equivalent to the original function code with the most significant bit set to logic one and appends a unique code in the data field of the response message which tells the master device what kind of error occurred or the reason for the error.



This function code 01, read coils code, is used to read from one to two thousand continuous registers for the status of coils in the slave device. The request PDU specifies the starting address of the slaves memory address of the first coil and the number of coils to read from the slave device.



The function code 02, read discrete inputs code, is used to read from one to two thousand continuous status of discrete inputs in a remote slave. The request PDU specifies the starting address of the slaves memory address of the first input and the number of coils to read from the slave device.

The function code 03, read holding registers code, is used to read the contents of a continuous block of holding registers in a remote slave. The request PDU specifies the starting register address, and the number of registers to read from the slave device.

The function code 04, read input registers code, is used to read from between 1 and 125 continuous input registers in a remote device. The request PDU specifies the starting register address and the number of registers.

The function code 05, write single coil code, is used to write a single output to either on or off in a remote slave device.

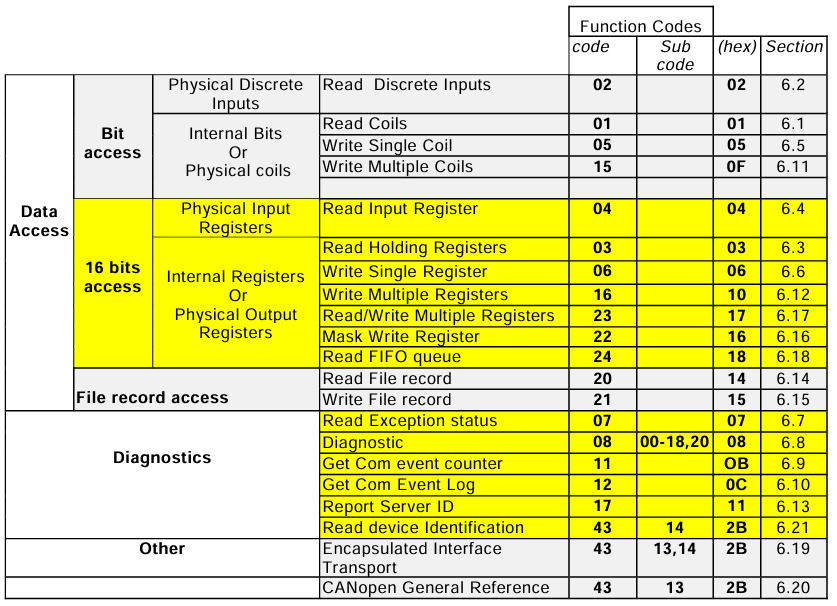
The function code 06, write single register code, is used to write a single holding register in a remote slave device. The request PDU specifies the address of the slave memory register address to be written to.

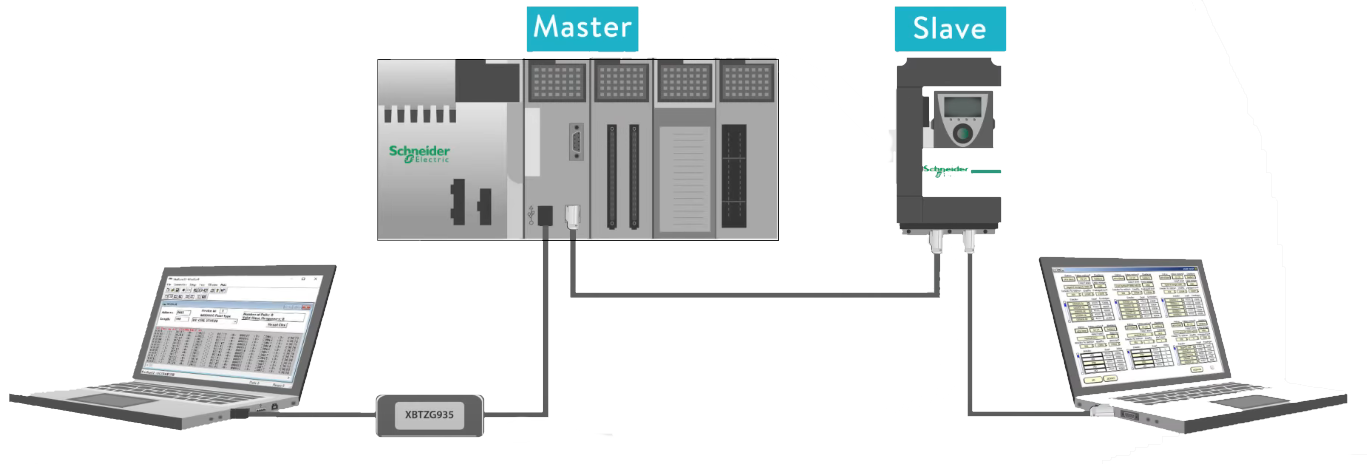
The function code 15, write multiple coils code, is used to force each coil in a sequence of coils to either on or off in a remote slave device. The request PDU specifies the coils memory address to be forced on or off.

The function code 16, write multiple registers code, is used to write a block of continuous registers between 1 and 123 registers, to a remote slave device.

These are the most common Modbus function codes. Review the Modbus specification for additional information.

<https://modbus.org/docs/Modbus_Application_Protocol_V1_1b3.pdf>





Communication with slave devices or master PLCs or computers can be accomplished with Modbus simulator software which runs on a PC. The connection can be either serial or Ethernet and in the form of a master or slave. The software will allow you to preform all of the Modbus protocol communication function codes to simply read or write to an existing slave. You can set up one PC to run the simulation software and another to run the master simulation software.

Connecting to a slave is first performed by setting the communication parameters for your serial comm port and then by entering the slaves address in the device ID field.

